Lesson 3 - Indexing

As with arrays in other programming languages, we can also use indexing to grab or edit certain elements from arrays. However, MATLAB has some extra features with indexing for more flexibility.

Basics of Indexing

Now for some reason, MATLAB's indexing does not start at 0; it actually starts at 1. If any of you have used Numpy, you may also know that Numpy uses 0-indexing and is row-major by default. This causes quite a few headaches when switching back and forth between the two.

```
somematrix = 1:16;

a1 = somematrix(5);
% Using 5 as the index gives you the 5th element, not the 6th
   element of the vector.

a2 = somematrix([4, 7, 13]);
% You can also get multiple elements with indexing.

a3 = somematrix([end - 2, end]);
% Matlab also has a little shortcut for accessing the end of a
   variable.
```

For 2D matrices, indexing is very similar to the 1D vector case (add another index).

```
matrix2 = reshape(1:16, 4, []);

b1 = matrix2(3, 2);
% Should give you 7 as an output

b2 = matrix2([1, 4], [1, 4]);
% This should return the submatrix with rows/columns 1 and 4. (
   In this case, we get [1, 13; 4, 16]).

b3 = matrix2(2, :);
% Instead of typing 1:end, Matlab also allows you to use : as a
    shortcut for indexing.

b4 = matrix2([end - 2, end]);
% For matrices, you can also access elements through linear
   indexing, which is effectively treating the matrix as a
   vector stored in column-major order.
```

You may find yourself switching between linear indexing and multiple indices in some situations. So, there are also functions to help switch between the two systems (namely sub2ind() and ind2sub()).

Changing, Adding, and Deleting Elements

Along with accessing elements with indexing, MATLAB also allows you to change whatever elements you want to access. Although it is usually more efficient to preallocate all the memory needed for a variable, it is also possible add and delete elements from matrices with indexing.

```
anothermatrix = reshape(1:16, 4, []);

anothermatrix(1, 3) = 1i;
% Changing the element in the 1st row, 3rd column
anothermatrix(5, :) = ones(1, 4);
% You can also change entire rows and columns
anothermatrix(:, 2) = [];
% Deleting rows/columns is done by assigning an empty array.

anothermatrix(10, 10) = 67;
% Adding an element beyond the sizes of the matrix will resize
   the matrix and fill the rest of the space with zeros.

anothermatrix([6, 7], [8, 9]) = 4 * ones(2, 2);
% Just as with indexing, you can change multiple elements at
   once.
```

Logical Indexing

The logical matrices from previous lesson also can also be used to combine indexing with some conditional restrictions for accessing elements. If I wanted to extract the corner elements for example,

```
shape = reshape(0:2:16, 3, [])

corners = logical([ 1, 0, 1
                    0, 0, 0
                    1, 0, 1]);

output = shape(corners);
% output should be [0; 4; 12; 16].
```

As the output of logical and relational operators are also logical matrices, we can use these to crate more complicated logical matrices for indexing.

```
A = reshape(1:24, 3, []);

lessthan5 = A < 5;
%{
Should return
[
   1, 1, 0, 0, 0, 0, 0, 0
   1, 0, 0, 0, 0, 0, 0, 0
   1, 0, 0, 0, 0, 0, 0, 0
]
%}

logicA = A <= 7 | A >= 13;
%{
Should return
[
   1, 1, 1, 0, 1, 1, 1, 1
   1, 1, 0, 0, 1, 1, 1, 1
   1, 1, 0, 0, 1, 1, 1, 1
]
%}
```

MATLAB also provides the find() function for retrieving linear indices under certain conditional restrictions.

```
x = linspace(-pi, pi, 1000);

higher = find(tan(x) > 0.5);
% Returns linear indices of all x where sin(x) > 0.5.

ex = exp(x);
closetotwo = find(mod(round(ex), 4) == 0 & round(ex) ~= 0);
% Also can string a lot of functions and conditions together
```